A Refinement Based Approach to Hybrid Systems: Basics

Richard Banach

School of Computer Science, University of Manchester, UK

Contents

1. System Development via Model Based Refinement

In model oriented development via refinement, we build models of the system, by specifying:

- the state (and I/O) space of a model
- the operations (or events) of a model: via eg.
  — transition systems, programming notations, etc.

Models can then be related pairwise by REFINEMENT. This usually involves a notion of correctness, relying on the substitutivity, of some **concrete** system behaviours for some **abstract** system ones, intended to help move closer to an implementation, and leading to sufficient conditions for refinement.

Effectively, it amounts to simulation.

2. Physical Systems in General, Control Systems in Particular

Many critical systems depend in an essential way on physical models — the systems are critical because they control critical plant.

Physical models are (almost invariably) governed and modelled by laws that depend on continuous mathematics.

In principle, such control could be implemented by analogue systems, but in practice, this is almost never done any more.

Continuous control is almost always implemented by digital controllers, which enact a discretization of the continuous control response with a very fast feedback.

Model based refinement techniques cannot capture the properties of the continuous to discrete modelling change.

Problems with Refinement

Applicability in the real world is often blocked.

1. Critical systems developers:
   - Need techniques giving very high assurance.
   - Understand and can benefit from the formal approach.
2. Often physical models are involved:
   The continuous/discrete modelling transition is not doable within refinement, restricting the scope of formal modelling. So the abstract model ends up in the discrete domain, bypassing much serious design.
3. Even for purely discrete applications:
   - The real world never starts from a blank sheet.
   - Real world complexity can prohibit 100% faithful models.
   - Management can impede adherence to refinement ideals.

3. Model Based Retrenchment

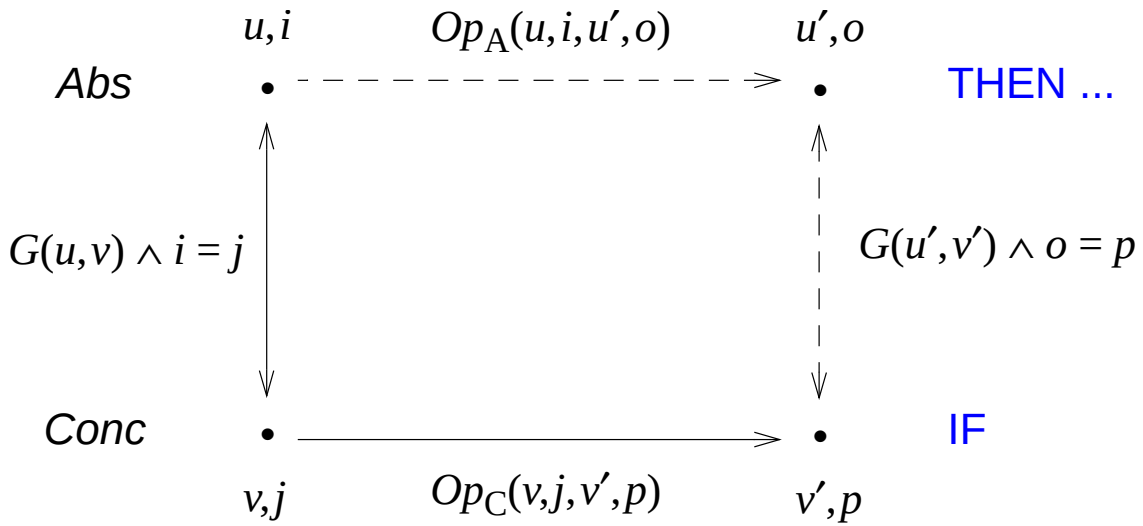
Inspired (in particular) by the obvious inappropriateness of a strict reading of refinement to any system based on physical laws, retrenchment was introduced to give refinement the little extra 'elbow room' it lacked for such applications.

In order to remain compatible with model based refinement, retrenchment was designed as a gentle weakening of the core proof obligation of refinement.

Specifically, additional relations were added to the core PO, to allow additional (inconvenient) facts about the application to be accommodated.

Obviously the addition of arbitrary elements to the PO destroys any connection with a preceding notion of correctness. Eventually it was realised that this was no problem — refinement-like notions of correctness ought not to be the province of retrenchment.

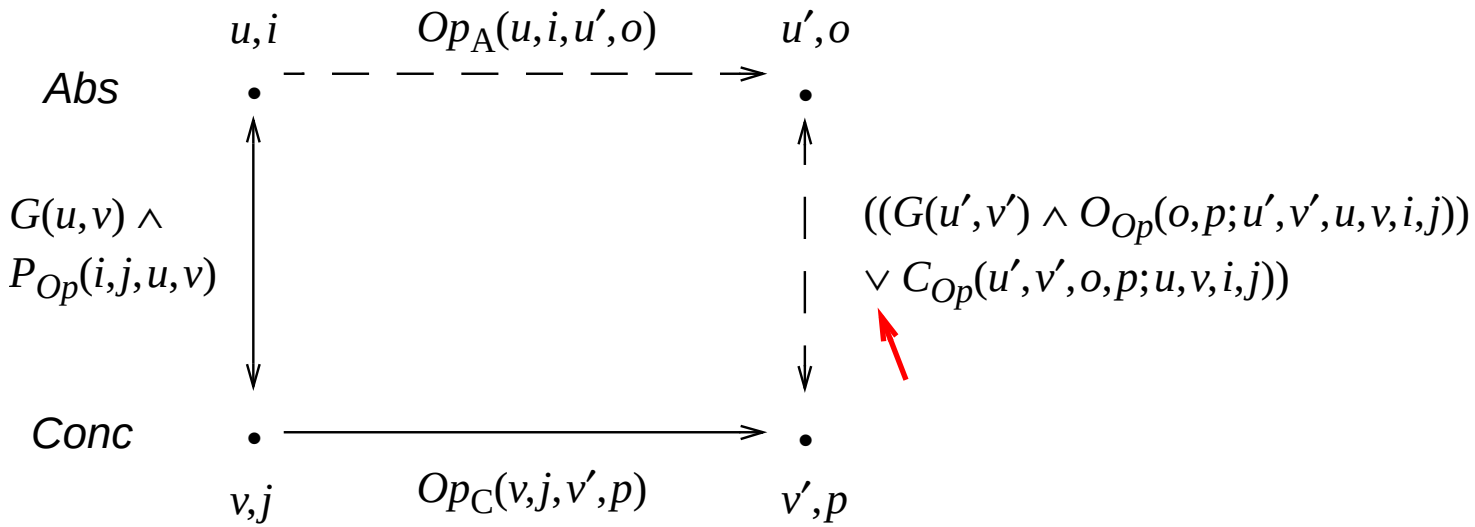
Refinement Fidelity PO



If the concrete system makes an Op_C move, then the move can be simulated by the abstract system making an Op_A move.

$$G(u,v) \wedge In_{Op}(i,j) \wedge Op_C(v,j,v',p) \Rightarrow$$
$$(\exists\, u', o \bullet Op_A(u,i,u',o) \wedge G(u',v') \wedge Out_{Op}(o,p))$$

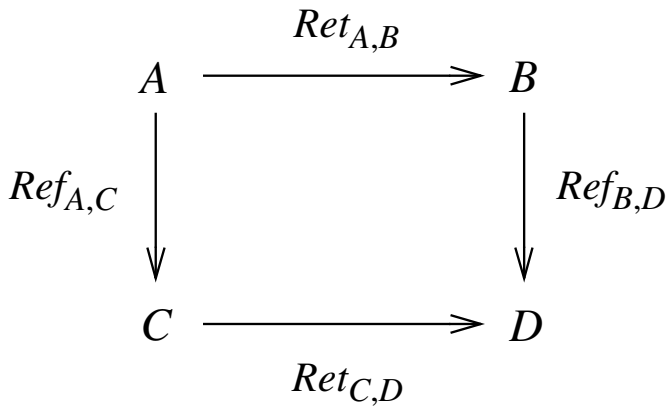
Retrenchment Fidelity PO



$$G(u,v) \wedge P_{Op}(i,j,u,v) \wedge Op_C(v,j,v',p) \Rightarrow$$
$$(\exists\, u', o \bullet Op_A(u,i,u',o) \wedge ((G(u',v') \wedge O_{Op}(o,p;u',v',i,j,u,v)) \vee$$
$$C_{Op}(u',v',o,p;i,j,u,v)))$$

The Tower Pattern

The main properties of retrenchment concern its interworking with refinement (so that its extreme permissiveness is utilised to the minimum). The *Tower Pattern* — retrenchments horizontal, refinements vertical:



Systems A, B, C, D form a compatibly commuting square.

Square completion theorems rebuild any missing one of A, B, C, D and its adjacent relations.

4. A Framework for Hybrid Systems

Integrating formal reasoning in discrete and continuous domains requires a suitable semantic framework, which:

- is expressive enough for continuous applications;
- defaults cleanly for discrete reasoning.

4. A Framework for Hybrid Systems

Integrating formal reasoning in discrete and continuous domains requires a suitable semantic framework, which:

- is expressive enough for continuous applications;
- defaults cleanly for discrete reasoning.

- Time is an interval $\mathcal{T}$ of the reals $\mathbb{R}$.
- There are **mode variables** (piecewise constant), and **pliant variables** (piecewise continuously varying).
- $\mathcal{T}$ partitions into a sequence of left-closed right-open intervals, $\langle [t_0 \ldots t_1), [t_1 \ldots t_2), \ldots \rangle$, such that (all) discontinuous changes take place at some boundary point t_i ... càdlàg.

In an interval $[t_i \ldots t_{i+1})$, the mode variables will be constant, but the pliant variables will change continuously, subject to:

I Zeno: there is a constant δ_{Zeno}, such that for all i needed, $t_{i+1} - t_i \geq \delta_{\mathsf{Zeno}}$.

II Limits: for every variable x, and for every time $t \in \mathcal{T}$, the left limit $\lim_{\delta \to 0} x(t - \delta)$ written $\overrightarrow{x(t)}$ and right limit $\lim_{\delta \to 0} x(t + \delta)$, written $\overleftarrow{x(t)}$ (with $\delta > 0$) exist, and for every t, $x(t) = \overleftarrow{x(t)}$.

III Differentiability: The behaviour of every pliant variable x in the interval $[t_i \ldots t_{i+1})$ is given by the solution of a well posed initial value problem $\mathcal{D}xs = \phi(xs, t)$. "Well posed" means $\phi(xs, t)$ has uniformly bounded Lipschitz constants (w.r.t. xs), and $\phi(xs, t)$ is measurable in t.

There are **mode transitions** ((any) variable can change discontinuously), and **pliant transitions** (pliant variables can change continuously). We say that a set of rules is **well formed** iff:

- Every enabled mode transition is feasible, i.e. has an after-state, and on its completion enables a pliant transition (but does not enable any mode transition).
- Every enabled pliant transition is feasible, i.e. has a time-indexed family of after-states, and EITHER:
  - (i) During the run of the pliant transition a mode transition becomes enabled. It preempts the pliant transition. ORELSE
  - (ii) During the run of the pliant transition it becomes infeasible: finite termination. ORELSE
  - (iii) The pliant transition continues indefinitely: nontermination.

A mode transition establishes the initial state.

5. Summary

Existing discrete event refinement formalisms (tacitly) assume isolated discrete events

... but are inadequate for encompassing continuous behaviour.

The least painful way to remedy this is to allow continuous behaviour in between the isolated discrete events.

- This basic picture is enough for model building and case studies.
- We can arrange the technical details to extend the original picture cleanly.